for said particular request, said cache is accessible to said particular request but not accessible to other requests, said cache is separate from said data store; and accessing a particular entry in said cache in response to said particular request calling for said Identity System to access said particular Identity System data entry in said data store, said particular entry in said cache corresponds to said particular Identity System data entry in said data store.

21. A method according to claim 20, wherein: said step of assigning is performed after said step of receiving and prior to said particular request calling for said Identity System to access said particular Identity System data entry in said data store.

22. A method according to claim 20, wherein: said particular request calling for said Identity System to access said particular Identity System data entry in said data store includes providing a query.

23. A method according to claim 20, wherein: said particular request calling for said Identity System to access said particular Identity System data entry in said data store includes requesting that particular Identity System data entry be written to said data store; and said step of accessing said particular entry in said cache includes writing particular entry in said cache.

26. The method of claim 20, wherein: said Identity System is integrated with an Access System.

29. One or more processor readable storage devices having processor readable code embodied on said processor readable storage devices, said processor readable code for programming one or more processors to perform a method comprising: receiving a particular request at an Identity System, said particular request calls for said Identity System to access a particular Identity System data entry in a data store; assigning a cache for said particular request, said cache is accessible to said particular request but not accessible to other requests, said cache is separate from said data store; and accessing a particular entry in said cache in response to said particular request calling for said Identity System to access said particular Identity System data entry in said data store, said particular entry in said cache corresponds to said particular Identity System data entry in said data store.

31. One or more processor readable storage devices of claim 29, wherein: said Identity System is integrated with an Access System.

33. An Identity System, in communication with a data store, that caches data associated with entries in said data store, comprising: a communication interface; one or more storage devices; and one or more processors in communication with said one or more storage devices and said communication interface, wherein said one or more processors perform a method comprising: receiving a particular request, said particular request calls for said Identity System to access a particular Identity System data entry in said data store, assigning a cache for said particular request, said cache is accessible to said particular request but not accessible to other requests, said cache resides in said one or more storage devices and is separate from said data store, and accessing a particular entry in said cache in response to said particular request calling for said Identity System to access said particular Identity System data entry in said data store, said particular entry in said cache corresponds to said particular Identity System data entry in said data store.

34. The Identity System according to claim 33, wherein said method further comprises: destroying said cache upon termination of said particular request.

35. The Identity System according to claim 33, wherein: said Identity System is integrated with an Access System.

36. The Identity System according to claim 33, wherein: said cache is not updated in response to entries in said data store being changed in response to requests other than said particular request.

Previous Doc Next Doc Go to Doc#

Freeform Search

Database: US Pre-Grant Publication Full-Text Database
US Patents Full-Text Database
US OCR Full-Text Database
EPO Abstracts Database
JPO Abstracts Database
Derwent World Patents Index
IBM Technical Disclosure Bulletins

Term: L4 AND AUXILIARY

Display: 50 **Documents in Display Format:** FRO **Starting with Number** 1

Generate: ○ Hit List ◉ Hit Count ○ Side by Side ○ Image

Search Clear Interrupt

Search History

DATE: Wednesday, October 04, 2006 Purge Queries Printable Copy Create Case

Set Name side by side	Query	Hit Count	Set Name result set
DB=USPT; PLUR=YES; OP=OR			
L7	L5 AND (REMOV$ NEAR AUXILIARY)	1	L7
L6	L5 AND (REMNOV$ NEAR AUXILIARY)	0	L6
L5	L4 AND AUXILIARY	4	L5
L4	L3 AND (FIRST NEAR ENTRY)	43	L4
L3	IDENTITY NEAR SYSTEM	894	L3
L2	L1 AND (FIRST NEAR ENTRY)	116	L2
L1	ENTITY NEAR SYSTEM	1763	L1

END OF SEARCH HISTORY

1. A method for caching data associated with entries in a data store accessed in response to a request assigned to a thread of execution, said method comprising: (a) determining that a first command arising from said request calls for access to a data store entry in said data store; and (b) accessing a first entry in a cache object in response to said first command, wherein said first entry corresponds to said data store entry, wherein said cache object is separate from said data store, wherein said cache object is associated with said thread of execution, and wherein said cache object is only accessed in response to commands arising from requests assigned to said thread of execution.

3. The method of claim 1, wherein said first command calls for writing a value to said data store entry, wherein said step (b) includes: (1) writing said value to said first entry in said cache object; and (2) writing said value to said data store entry.

4. The method of claim 3, wherein a second command arising from said request calls for a query from said data store entry, wherein said method further includes: (d) retrieving said value from said first entry in said cache object.

9. One or more processor readable storage devices having processor readable code embodied on said processor readable storage devices, said processor readable code comprising: code that receives a request and assigns said request to a thread of execution, said request relates to one or more entries in a data store code that determines that a first command arising from said request calls for access to a data store entry in said data store; and code that accesses a first entry in a cache object in response to said first command, said cache object is separate from said data store, wherein said first entry corresponds to said data store entry, wherein said cache object is associated with said thread of execution, and wherein said cache object is only accessed in response to commands arising from said request.

13. A system for caching data associated with a data store, said system comprising: means for receiving a request and assigning said request to a thread of execution, said request relates to one or more entries in said data store; means for determining that a first command arising from said request calls for access to a data store entry in said data store; and means for accessing a first entry in a cache object in response to said first command, wherein said first entry corresponds to said data store entry, wherein said cache object is associated with said thread of execution, and wherein said cache object is only accessed in response to commands arising from requests assigned to said thread of execution.

16. The system of claim 13, wherein: said one or more processors are part of an integrated Identity System And Access System; and said data store entry in said data store includes Identity data.

18. A method for caching data associated with entries in a data store accessed in response to a request assigned to a thread of execution, said method comprising the steps of: (a) determining that a first command arising from said request calls for access to a data store entry in said data store; (b) accessing a first entry in a cache object in response to said first command, wherein said first entry corresponds to said data store entry, wherein said cache object is not stored in said data store; wherein said cache object is associated with said thread of execution, wherein said thread of execution contains a thread local storage with a pointer to said cache object, wherein said cache object is only accessed in response to commands arising from said request assigned to said thread of execution, wherein said first command calls for writing a value to said data store entry, and wherein said step (b) includes the steps of: (1) writing said value to said first entry in said cache object, and (2) writing said value to said data store entry; (c) retrieving said value from said first entry in said cache object in response to a second command arising from said request, wherein said second command calls for a query from said data store entry; and (d) destroying said cache object in response to said request being completed.

20. A method for caching data associated with entries in a data store, comprising: receiving a particular request at an Identity System, said particular request calls for said Identity System to access a particular Identity System data entry in said data store; assigning a cache

L5: Entry 1 of 4 File: USPT Jan 6, 2004

DOCUMENT-IDENTIFIER: US 6675261 B2
TITLE: Request based caching of data store data

Brief Summary Text (7):
Identity Systems have become more popular with the growth of the Internet and the use of networks and other information technologies. In general, an Identity System provides for the creation, removal, editing and other management of identity information stored in various types of data stores. The identity information pertains to users, groups, organizations and/or things. For each entry in the data store, a set of attributes is stored. For example, the attributes stored for a user may include a name, address, employee number, telephone number, email address, user ID and password. The Identity System can also manage access privileges that govern the subject matter an entity can view, create, modify or use in the Identity System.

Brief Summary Text (8):
Identity System users direct the operation of the Identity System by submitting requests that call for an Identity System response, such a searching and viewing a user's profile. Requests frequently require the Identity System to repeatedly access the same entries in the Identity System's data store. For example, a request may cause the Identity System to load data into a data store entry and later retrieve the newly loaded data multiple times for performing different functions. This can occur when a client provides identification information that is stored in a data store entry. The request may retrieve this information on multiple occasions for forwarding to servers or applications accessed by the client request.

Brief Summary Text (12):
Each client request is assigned to a thread of execution in an Identity Server within an Identity System. A cache object is associated with the thread of execution for caching data store entry accesses arising from the request. In one implementation, the thread of execution contains a thread local storage with a pointer to the cache object. Employing the cache object to maintain frequently accessed data store entries reduces the number of data store accesses required to service the request--speeding request processing time and freeing data store bandwidth.

Brief Summary Text (13):
In one embodiment of the present invention, an Identity Server receiving a request interprets a command in the request to call for an access to a data store entry. In response to the command, the Identity Server accesses a first entry in the cache object that corresponds to the data store entry called for in the command. If the cache object does not include an entry corresponding to the data store entry, the Identity Server creates and loads a corresponding entry in the cache object.

Brief Summary Text (16):
In further embodiments of the present invention, request based caching is employed in processing environments other than an Identity System. Request based caching can have broad applicability to enhance the performance of many different server based systems.

Drawing Description Text (7):
FIG. 6 is a flow chart describing one embodiment of a process for accessing the Identity System.

Drawing Description Text (33):

FIG. 32 is a flow chart describing an overview of an exemplar process for adding and removing auxiliary classes.

Drawing Description Text (34):
FIG. 33 is a flow chart describing one embodiment of a process for removing auxiliary classes.

Drawing Description Text (35):
FIG. 34 is a flow chart describing one embodiment of a process for adding auxiliary classes.

Detailed Description Text (2):
FIG. 1 depicts an access management system, which provides identity management services and/or access management services for a network. The identity management portion of the system (hereinafter "the Identity System") manages identity profiles, while the access management portion of the system (hereinafter "the Access System") provides security for resources across one or more Web Servers. A key feature of one embodiment of this system is the centralization of the repositories for policies and user identity profiles, while decentralizing their administration. That is, one embodiment of the system centralizes the policy and identity repositories by building them on a directory service technology. The system decentralizes their administration by hierarchy delegating administrative roles. Although the system of FIG. 1 includes an Identity System and an Access System, other embodiments may only include an Identity System or only include an Access System.

Detailed Description Text (3):
FIG. 1 is a block diagram depicting one embodiment for deploying an integrated Identity System and Access System. FIG. 1 shows web browsers 12 and 14 accessing Web Server 18 and/or Web Server 20 via network 16. One example of a network is the Internet. In one embodiment, web browsers 12 and 14 are standard web browsers known in the art running on any suitable type of computer. FIG. 1 depicts web browsers 12 and 14 communicating with Web Server 18 and Web Server 20 using HTTP over the Internet; however, other protocols and networks can also be used.

Detailed Description Text (11):
The Identity System includes Web Pass 38, Identity Server 40 and Directory Server 36. Identity Server 40 manages identity profiles. An identity profile is a set of information associated with a particular entity (e.g. user, group, organization, etc.). The data elements of the identity profile are called attributes, which are discussed in more detail below. An attribute may include a name, value and access criteria. The Identity Server includes three main applications, which effectively handle the identity profiles and privileges of the user population: User Manager 42, Group Manager 44, and Organization Manager 46. User Manager 42 manages the identity profiles for individual users. Group Manager 44 manages identity profiles for groups. Organization Manager 46 manages identity profiles for organizations. Identity Server 40 also includes Publisher 48, an application that enables entities to quickly locate and graphically view information stored by Directory Server 36. In one embodiment, Web Pass 38 is a Web Server plug-in that sends information back and forth between Identity Server 40 and the Web Server 20, creating a three-tier architecture. The Identity System also provides a Certificate Processing Server (not shown in FIG. 1) for managing digital certificates.

Detailed Description Text (15):
The Identity System also provides for self-registration. User Manager 42 enables an individual to self-register in situations when it's appropriate. User Manager 42 then authorizes delegated administrators to verify the individual's information and approve or deny the registration requests. In one embodiment, self-registration is defined by a customizable, multi-step workflow. This concept is discussed below.

Detailed Description Text (18):
The third application in the Identity System, Organization Manager 46, streamlines the management of large numbers of organizations within an e-business network, including partners, suppliers, or even major internal organizations such as sales offices and business units. Certain infrastructure security and management operations are best handled--or can only be handled--at the highest organizational unit level rather than at the individual or group level. Like User Manager and Group Manager, this application relies on multi-step workflow and

delegation capabilities. Organization Manager handles the following administrative tasks: (1) organization lifecycle management, whereby companies can create, register, and delete organizations in their systems using customizable workflows; (2) maintenance of organization profiles on an attribute-by-attribute basis through self-service, delegated administration and system-initiated activities; (3) organization self-registration, whereby organizations such as business partners, customers and suppliers can self-generate a request to be added to the e-business network; and (4) creation of reusable rules and processes through multi-step workflows.

Detailed Description Text (20):
With the system of FIG. 1 deployed, Web Server 18 (enabled by Web Gate 28, Access Server 34, and Directory Server 36) can make informed decisions based on default and/or specific rules about whether to return requested resources to an end user. The rules are evaluated based on the end user's identity profile, which is managed by the Identity System. In one embodiment of the present invention, the general method proceeds as follows. An end user enters a URL or an identification of a requested resource residing in a protected policy domain. The user's browser sends the URL as part of an HTTP request to Web Server 18. Web Gate 28 intercepts the request. If the end user has not already been authenticated, Web Gate 28 causes Web Server 18 to issue a challenge to the browser for log-on information. The received log-on information is then passed back to Web Server 18 and on to Web Gate 28.

Detailed Description Text (37):
Database proxy 154 encapsulates the supporting agent objects for the particular operation. It also acts as a storage area where input parameters and output results are stored. Each database proxy object exposes its methods and input parameters. These parameters include search base, object class, auxiliary class, filter, search scope, attributes and entry. After a database client sets all the parameters, the client calls the execute method of the proxy to invoke the database operation. The client then calls the database proxy GetResults method to retrieve the operations results.

Detailed Description Text (50):
FIG. 5 shows a hierarchical tree. Some organizations employ fat or flat trees for ease of maintenance. A flat directory tree is a directory information tree that does not have any hierarchy. All of the nodes are leaf nodes (nodes without any child nodes). A fat directory tree is a tree that has a large number of nodes at any given level in a directory information tree. One advantage of a fat or flat tree is user maintenance. For example, if an employee moves to a new group, the node must be moved to a new container if the tree is not flat or fat. By moving the node to a new container, the distinguished name for the node changes and all certificates become void. One drawback of flat or fat trees is that the organization loses the benefits of having a logical directory, such as using the logical directory to determine who has access to which nodes. To remedy this, the Identity System includes partition support for fat and flat tree directories using filters. From a configuration page, an attribute can be configured to be accessible (read, modify, etc.,) based on a two part filter. The first component in the filter identifies a top node in the directory. The filter will only apply to those entities at or below that top node. The second component of the filter is an LDAP filter which defines who can access the attribute. This two component filter can be applied on an attribute by attribute basis.

Detailed Description Text (51):
There are many ways for an entity to access and use the Identity System. In one embodiment, the entity can access the Identity Systems services using a browser. In other embodiments, XML documents and API's can be used to access the services of the Identity System. For example, an entity can use a browser by pointing the browser to Identity Server 40. The user will then be provided with a login page to enter the user's ID, password, type of user and application requested (optional). Upon filling out that information, the user will be authenticated and authorized (by the Access System) to use the Identity System, as described below. Alternatively, the Access System can be bypassed (or there may be no Access System) and the Identity System authenticates the user.

Detailed Description Text (52):

FIG. 6 is a flowchart, which describes a process of entering the Identity System. In step 300 the user requests access to the Identity System. For example, the user can point a browser to Identity Server 40. After being provided with a login page, the user fills in the login information, and that information is sent back to the system. If there is an Access System, as described below, then the user will be authenticated and authorized by the Access System. After authorization, the request will be redirected from the Access System to Web Server 20 (see FIG. 1). If there is no Access System, or if the Access System is not providing authentication and/or authorization services, the browser can initially be pointed directly to Web Server 20. Other alternatives can also be supported. Upon the request being sent to Web Server 20, the request will be intercepted by Web Pass 38 in step 302. In step 304, it is determined whether there is an Identity System UidCookie. The UidCookie is stored on the user's system and can be provided with the request.

Detailed Description Text (53):
FIG. 7 depicts an example of UidCookie 360. A cookie is information that a web page, system or resource stores on a client device. In some embodiments it can represent information about the user, regardless of where it is stored and in what format. This cookie includes at least three components: Uid 362, IP address 364 and timestamp 366. Uid 362 stores the user identification for the entity trying to access the Identity System. IP address 364 is the IP address of the machine that the user is currently using. Timestamp 366 indicates the time that the cookie was initially created. Some embodiments use timestamp 366 to limit the life of the cookie. Some embodiments do not use timestamp 366. In one embodiment, the cookie is encrypted.

Detailed Description Text (54):
If, in step 304, it is determined that a valid UidCookie exists, then, in step 306, the user is given access to the Identity System application requested. The Uid from the cookie is used as the user identification upon entering the Identity System. If the valid UidCookie does not exist (step 304), then it is determined whether a user identification was received in a header variable. In one embodiment using an integrated Access and Identity System, a user's request to access the Identity System will be authenticated and authorized by the Access System. After authentication and/or authorization, the HTTP request will be redirected to the Identity System. This redirected HTTP request will include a header variable labeled as "userAuth." The data associated with this header variable will indicate the user identification for the user. If the user identification was in a header variable then a UidCookie is created in step 310 and that user identification is added to the UidCookie. Subsequent to step 310, the user is provided access to the Identity System in step 306.

Detailed Description Text (55):
If the user identification was not in a header variable, then the system attempts to authenticate the user in step 312. That is, the user's user name and password provided by the login page are used to access Directory Server 36 in order to authenticate the user. More information about authentication is described below. If the user is properly authenticated, then a UidCookie is created in step 316. During an authentication process, the user's ID and password were used to access the user's identity profile in Directory Server 36. That identity profile will include a user identification, which is added to the UidCookie in step 316. In one embodiment, the user identification is the user's distinguished name. In step 318, the user is provided access to the Identity System. If the user was not properly authenticated, then the user is denied access to the Identity System in step 320.

Detailed Description Text (56):
As discussed above, when requesting access to the Identity System, the user selects which of the Identity System applications (User Manager 42, Group Manager 44, Organization Manager 46 or Publisher 48) the user wishes to access. In one embodiment, the login page for the Identity System will request an ID, a password, an indication of the application requested and an indication of a role (discussed below). After appropriate authentication and authorization, the user is provided with a home page for User Manager 42, a home page for Group Manager 44, a home page for Organization Manager 46 or a home page for Publisher 48, depending upon which application was selected by the user. From the home page, the user can access the various services of the application.

Detailed Description Text (71):
As discussed above, when an entity logs into the Identity System, the entity indicates the entity's role. There are at least six roles: System Administrator, Master Identity Administrator, Master Access Administrator, Delegated Access Administrator, Delegated Identity Administrator and End User. The System Administrator can perform all Access System configuration tasks and all Identity System configuration tasks. The Master Identity Administrator can configure access controls, attribute access controls, new user services, workflow definitions, setting the search base, delegating rights, expanding dynamic groups, and setting container limits. The Master Access Administrator can configure a web gate, configure an access server, create host identifiers, configure users, set-up policies and policy domains, and delegate rights. The Delegated Identity Administrator is an administrator who has been delegated rights from the Master Identity Administrator. The Delegated Access Administrator can be delegated rights from a Master Access Administrator. An End User cannot perform configuration functions. There can also be a delegated admin who can create/delete users, add/remove users to/from groups, process workflow steps, etc.

Detailed Description Text (73):
One right that an administrator has and which can be delegated to a Delegated Administrator is the proxy right. The proxy right for person A allows person A to choose another person (e.g. person B) to be a proxy for person A during a period of time. For example, if a Delegated Administrator (or other administrator) is going on vacation, or will otherwise be unavailable to perform its administrative duties, that Delegated Administrator can identify another person (or persons) who can be a proxy for that Delegated Administrator. While person B is being a proxy for person A, person B has all the rights and privileges of person A within the Identity System. Person B does not have the rights of person A in the Access System. Thus, the Identity System will see person B as person A, but the Access System will see person B as person B.

Detailed Description Text (77):
In step 668, the UidCookie on the user's machine is edited by changing Uid 362 to equal the user identification for the person being proxied. In step 670, the user now operates as the person being proxied in the Identity System. Because the Uid in the Cookie identifies the person being proxied, the Identity System treats the user as the person being proxied. However, the UidCookie is only used by the Identity System, so only the Identity System treats the person as the person being proxied. The Access System uses a different cookie (described below), and the Access System's cookie is not edited. Therefore, the Access System treats the user as himself or herself and not as the person being proxied. While being a proxy, the user has all the rights and privileges as the person being proxied. In one embodiment, the process of FIG. 15 is performed without the user providing or knowing the password for the person being proxied and. therefore, without authenticating the password and ID for the person being proxied.

Detailed Description Text (78):
In one embodiment, step 670 includes receiving a request from the user (e.g. the entity who is the proxy) to access a service of the Identity System. In response, the system will access the Uid in the cookie, and use that Uid to access attributes, group memberships and organizations memberships for the identity profile of the person being proxied. Based on those attributes, the user will or will not be provided access to the requested service.

Detailed Description Text (80):
A lot of the tasks that are performed in the Identity System are accomplished using workflows. A workflow is a predefined set of steps that perform a specific task, where information or tasks are passed between participants and programs according to a defined set of rules. One embodiment of the present invention supports the following types of workflows: create object; delete object; change the value of attributes; and certificate issuance, revocation and renewal. In one embodiment of the present invention, a user is required to create a workflow to create or delete an object, change the value of an attribute or implement certificates. Workflows ensure that an organization's guidelines for performing a task are met. Workflows can be defined in the User Manager, Group Manager or Organization Manager. A workflow can be used only in the application (e.g. User Manager) in which it was created. Each workflow has two or more steps, including one to start the action and one to implement or commit it. Each step can

contain an action, send e-mail notifications to selected persons and start the next step if its entry conditions are satisfied. A workflow is associated with a portion of the directory tree. This allows an entity to have its organizations and partners enforce different workflows. Workflows can be stored in Directory Server 36.

Detailed Description Text (96):
The workflow that initiates the subflow is referred to as the parent workflow. A workflow can be both a parent workflow to a first workflow and a subflow to a second workflow. The parent workflow may or may not wait for the subflow, as defined in the workflow creation. Consider the following example, a company uses a first workflow to create new users for the Identity System and add the new user's identity profile to the directory. As part of its process, the new user workflow obtains the new user's telephone number. The obtaining of the new user's telephone number is accomplished by performing a new telephone number workflow. In this example, the new telephone number workflow is initiated by a step in the new user workflow. Therefore, the new telephone number workflow is a subflow of the new user workflow. In one alternative, the new telephone number workflow can also call a subflow, for example, to get a new telephone line connected and operational. This, second subflow can also call a subflow, and so on. There can be many levels of nesting of subflows. Additionally, a parent workflow can have many subflows.

Detailed Description Text (108):
The cross application workflow uses a pre and post processing feature of the integrated Identity System and Access System. The pre and post processing allows third parties to extend the base of functionality of the system by providing custom actions based on specific defined events. The base elements of pre and post processing are called events. Events occur any time the user interacts with the system. Events can be as simple as adding, modifying or deleting an object or could be as complex as a specific step within a workflow process.

Detailed Description Text (112):
In one embodiment, the integrated Access and Identity System accepts XML document inputs that are encapsulated in a SOAP envelope using HTTP protocol requests. The XML document contains the necessary parameters and authentication information for carrying out the request. The request is sent to an appropriate URL for the desired application. The Identity System provides the desired application's response to the client program as an output XML document.

Detailed Description Text (126):
In step 1106, the set of groups that the user is a static member of and the set of groups that the user is a dynamic member of are combined to determine the set of groups in which the user is either a dynamic or static member. In step 1108, the final set of groups G.sub.t is initialized to the set of groups in which the user is either a static member or dynamic member. For each group in which the user is a static or dynamic member, the system calls the function Find_Containing_Groups (step 1110). The results of the function are added to the set G.sub.t. In step 1114, the resulting set G.sub.t is reported as an identification of all the groups in which the user is either a static, dynamic or nested member. The resulting set can be reported in various ways including reporting the groups in a GUI for the user (e.g. a tree on its side), reporting the groups to the user in a non-graphical format, storing a list of the groups in a file, providing identifications of the groups to another process, etc. In one example, the access system requests that the Identity System determine a user's groups so that the access system can authorize a user to access a resource based on membership in a particular group.

Detailed Description Text (147):
Another feature of Group Manager 44 is the ability to dynamically modify groups during run time. This feature is based on attaching auxiliary object classes to structural object classes. A structural object class can be instantiated to create a group such that for each entry in the directory there is only one structural object class. The structural object class cannot change after the object has been instantiated and is being used. One or more auxiliary object classes can be attached to any structural object class in a directory. The structural object class defines a set of attributes. The auxiliary object class also has a set of attributes. When an auxiliary object class is attached to an object class, the attributes of the auxiliary class are added to the object. Once instantiated, a structural object class cannot be modified or removed; auxiliary object classes, however, can be added or removed. Group manager 44 provides

the user with the ability to add or remove auxiliary object classes on the fly using a GUI.

Detailed Description Text (148):
Prior identity systems allow for the addition of auxiliary classes to structural classes upon creation of the object. The present invention allows for auxiliary classes to be added and removed subsequent to object creation. That is, dynamically, an existing object class can have additional attributes added to the group object or removed from the group object by adding or removing auxiliary classes.

Detailed Description Text (149):
When creating a group, an administrator (or other user with sufficient privileges) is provided with a graphical user interface that lists all possible attributes that can be included in the group profile. Some of these attributes are part of structural object class, while others are part of auxiliary object classes (or auxiliary object class schema). If the user selects attributes from an auxiliary class, then those auxiliary classes are added to the object upon creation of the object. After the group is created, various attributes can be populated with data values. Subsequent to this time, attributes that are associated with auxiliary classes can be removed or added to the group. In addition to adding flexibility to defining which attributes are associated with a group, the present invention allows for bulk deletion of attributes. Simply removing the auxiliary object class from the group entry will automatically delete all attributes of the removed auxiliary object class.

Detailed Description Text (150):
FIG. 32 is a flowchart describing an overview of the process for adding and removing attributes to a group during run time. In step 1398, a group is created. This step includes determining which attributes to include in the group definition. Based on the attributes chosen, a structural class and the appropriate auxiliary classes are added to the group. In one implementation, the group is created by instantiating the appropriate classes to create a group object representing the group identity profile. In one embodiment, a group can be created that has an auxiliary class, but no attributes of that auxiliary class. The system can use a workflow to create the group and the workflow knows which auxiliary classes to use. The arrow from step 1398 to step 1400 is depicted as a doted line to indicate that time and other steps pass before step 1400 is performed. That is, step 1400 is performed after a group has been created and, possibly, after the various attributes have been populated with data. In step 1400, Group Manager 44 receives a request to modify the existing group. This can happen from Configure tab 440. Alternatively, while viewing a group, Group Manager 44 will display a "modify group" button. Selecting that button allows the user to request a modification to the group being viewed, if the user has sufficient privileges. In step 1402, Group Manager 44 provides a list of auxiliary classes that can be added or removed from the existing group. In an alternative embodiment, Group Manager 44 provides a list of attributes to add or remove, with each of the attributes being associated with auxiliary classes. The auxiliary classes and/or attributes to be added or removed are reported to the user via a graphical user interface. Next to each class (or each attribute) is a check box. The user can check the check box to indicate that the class (or attribute) should be added. The user can uncheck check box to indicate that the class (or attribute) should be removed. In step 1404, the selection of classes (or attributes) to be added and removed are received by Group Manager 44 from the graphical user interface and stored. In step 1406, those auxiliary classes selected to be removed are then removed from the group object including removing those attributes from the group object. In step 1408, the auxiliary class selected to be added and their associated attributes are added to the group object. After step 1408, the group can be used as any other group; for example, a user can be authorized to access a resource based on attributes of or membership in a group.

Detailed Description Text (151):
FIG. 33 is a flowchart describing the process for removing auxiliary classes and their associated attributes from an object. In step 1430, Group Manager 44 selects one of the classes that have been marked for removal. In step 1432, Group Manager 44 determines which attributes are associated with that selected auxiliary class. The attributes identified in step 1432 do not include attributes that are part of a class that is not being removed. In step 1434, those attributes that are determined in step 1432 are removed from the group object. When the

attributes are removed, all data stored in those attributes is deleted. In step 1436, the actual auxiliary class is removed from the group object. In step 1438, all auxiliary classes that are superior classes to the currently selected auxiliary class (see step 1430) are removed from the group object. In many instances, the auxiliary classes are part of an object oriented hierarchy where auxiliary classes can be subclasses of other classes (called superior classes). A subclass inherits from the superior class. In many cases, a particular auxiliary class may have a superior class, which has a superior class, which has a superior class, and so on. Thus, the chain of superior classes from the auxiliary class will go all the way up the tree to the root class. Therefore, some auxiliary classes will have many superior classes. All of the superior classes for a particular auxiliary class are removed when that auxiliary class is removed. Step 1436, however, does not remove a superior class, if that superior class is also superior to another auxiliary class that is part of the object and is not being removed. There is no need to remove the attributes of the superior classes because all those attributes have been inherited by the auxiliary class and already removed in step 1434. In step 1440, it is determined whether there are any more auxiliary classes to be removed. If there are more auxiliary classes to be removed, then the method loops to step 1430. If there are no more auxiliary classes to remove, then the process is complete. Note that some directories do not allow for the modification of the object class attribute; therefore, in those cases, only the attributes are removed.

Detailed Description Text (152):
FIG. 34 is a flowchart describing a process for adding to the group object those auxiliary classes that have been marked for addition. In step 1460, Group Manager 44 chooses an auxiliary class for adding to the group object from those auxiliary classes marked for addition. In step 1462, the chosen auxiliary class is added to the group object. In step 1464, all superior classes of the auxiliary class chosen in step 1460 that are not already part of the group object are added to the group object. In step 1466, all of the attributes from the auxiliary class selected in step 1460 are added to the group object. In step 1468, it is determined whether there are any more auxiliary classes to add. If there are more auxiliary classes to add, then the method loops back to step 1460. If there are no more auxiliary classes to add, then the method of FIG. 34 is completed.

Detailed Description Text (153):
The ability to add or remove from an existing group at runtime provides greater flexibility in defining the content for groups. Furthermore, the removal of an auxiliary class provides a means to bulk delete a set of attributes because removing an auxiliary class will, in one embodiment, delete all attributes for the removed class. Finally, the ability to add or remove from an existing group provides for less coupling between a group schema and group entries. For example, if the schema changes such that a group auxiliary class is removed, only those group entries that have that auxiliary class need to be updated.

Detailed Description Text (154):
The Identity System also includes an "Advanced Group" auxiliary object class that contains the attributes necessary to implement some of the unique functionalities described above. Administrators can attach the "Advanced Group" to a group in order to provide values for attributes that control features such as Subscription/Unsubscription and Dynamic Membership. In one embodiment, the "Advanced Group" consists of one auxiliary class that includes the attributes listed below. In another embodiment, the "Advanced Group" consists of a plurality of classes.

Detailed Description Text (174):
XML data registry 1670 contains registration files. Each registration file corresponds to at least one program or peripheral programs listed in program service 1660. Each registration file contains information necessary for structuring the output of a program's result. Identity Server 40 maintains a set of XML templates 1672, XML schemas 1674, and XSL stylesheets 1676. Each registration file in data registry 1670 contains a pointer to an XML template, an XML schema and XSL stylesheet. The application of templates and stylesheets will be explained below in greater detail. Schemas provide information to Identity System users for establishing display characteristics.

Detailed Description Text (196):
In one embodiment, Identity Server 40 obtains attribute display characteristics from directory entries in Directory Server 36. Each Directory Server entry corresponds to a different attribute type. For each attribute, Identity Server 40 locates a corresponding directory entry, which provides the attribute's display characteristics. In one such embodiment, a system administrator creates all the display attribute directory entries when Identity System 40 is configured. In alternate embodiments of the present invention, the directory entries are replaced by tables, data structures, or other means that relate display characteristics to attributes so the display characteristics can be obtained by Identity Server 40.

Detailed Description Text (204):
Requests for data received by the Identity System frequently require repeated access to the same entries in Directory Server 36. Continually retrieving this information through Directory Server 36 slows operation and wastes server bandwidth. Therefore, Identity Server 40 provides each active request with a cache to reduce the number of data store accesses.

Detailed Description Text (211):
As described above, clients submit requests to the Identity System asking for information on requesting tasks to be performed. These requests can be submitted via HTTP, XML documents, or other means. In some embodiments of the present invention, multiple Identity Servers are employed to increase the throughput of the Identity System. In such embodiments, requests are assigned to Identity Servers so as to balance the load of each Identity Server. In some instances a request may execute a function that requires a primary Identity Server handling the request to communicate with another Identity Server.

Detailed Description Text (221):
After executing the local operation or if no local operation is required, management service 1910 opens a message channel for providing the remote request to remote Identity Server 1902 (step 1966). Management service 1910 then issues the remote request to remote Identity Server 1902 (step 1968). In the embodiment shown in FIG. 46, management service 1910 opens up a communication channel with Identity Server 1902 and provides the remote request to server 1902. In alternate embodiments, however, more than two Identity Servers are employed in the Identity System. In such embodiments, local Identity Server 1900 opens message channels with all the other remote Identity Servers and issues the remote request to them.

Detailed Description Text (227):
Embodiments of the present invention provide for establishing different sets of criteria for obtaining a certificate. For example, a high level person in an organization may have great need for access to confidential corporate information. The corporation may wish to issue this person a certificate without any more than a mere request being filed. On the other hand, entry level employees at a corporation may have very little need for access to confidential information. The corporation may wish to have the entry level person's manager approve the issuance of a certificate. One embodiment of the integrated Access and Identity System of the present invention incorporates certificate management into the workflow process so different standards for certificate management can be applied among various entities. In one implementation, different certificate enrollment, renewal, and revocation workflows can be defined for different types of system users.

Detailed Description Text (229):
The integrated Access and Identity System of the present invention also includes Certificate Processing Server 2076, which is in communication with Identity Server 40 to communicate with certificate registration module 2072. Certificate Processing Server 2076 issues certificate signing requests to Certificate Authority 2084, which is external to the integrated Access and Identity System and in communication with Certificate Processing Server 2076. Certificate Authority 2084 is typically a third party vendor that provides certificates, including pairs of public and private keys for attachment to the certificates. One example of a third party certificate provider is Verisign. Certificate Processing Server 2076 is also in communication with signing device 2078. Signing device 2078 digitally signs certificate signing requests before they are issued to Certificate Authority 2084. Digitally signing certificate signing requests heightens the level of security in the connection between Certificate Processing

Server 2076 and Certificate Authority 2084. In one embodiment of the present invention, certificate registration module 2072 communicates with Certificate Processing Server 2076 via a secure SSL socket connection and Certificate Processing Server 2076 communicates with Certificate Authority 2084 via a secure SSL connection to enhance system security.

Detailed Description Text (232):
Certificate registration module 2072 proceeds with certificate enrollment in accordance with the workflow by retrieving information (step 2122). Examples of the information retrieved include information from the user's identity profile and information from entities associated with the user. Examples of entities associated with the requesting user include the requesting user's manager who also has an identity profile in the Identity System and can be contacted by Identity Server 40.

Detailed Description Text (237):
Once a certificate has been issued it is typically valid for a predetermined period of time, such as one year. After the time period expires, the certificate holder must renew the certificate. In one embodiment of the present invention, the certificate holder renews the certificate by submitting a certificate renewal request to Identity Server 40. This request is handled by certificate registration module 2072 in essentially the same manner as described above for certificate enrollment. The same process is applicable, because the renewal of a certificate is essentially the same as enrollment. When a certificate is renewed, Certificate Authority 2084 generates a new private key-public key pair, in essence creating a new certificate without increasing the total number of certificates issued to the Identity System. The only difference is that Certificate Processing Server 2076 informs Certificate Authority 2084 that a certificate is to be renewed, as opposed to a new certificate being issued.

Detailed Description Text (253):
As described above, Identity Server 40 maintains public copies of certificates in certificate data store location 2082. Identity System users issue requests to Identity Server 40 to export or display the certificates. In one embodiment of the present invention, the Identity System maintains real time status information about the certificates, so users are not unknowingly importing or viewing expired certificates. Maintaining this status information is beneficial, because certificate status is a dynamic value that cannot typically be provided in a certificate field.

Detailed Description Text (257):
FIG. 59B illustrates a sequence of steps carried out by Identity Server 40 to export a certificate in one version of the present invention. Identity Server 40 receives a user request via Web Server 20 to export a certificate from certificate data store location 2082 (step 3420). Identity Server 40 determines whether to check the status of the requested certificate (step 3422). In one implementation, Identity Server 40 makes this determination by querying a parameter field in the Identity System. This parameter field can be set by a system administrator during system configuration.

Detailed Description Text (258):
If a status check is not required, Identity Server 40 exports the requested certificate to the user via Web Server 20 (step 3434). Otherwise, Identity Server 40 determines whether a real time status check of the certificate is required (step 3424). Identity Server 40 also makes this determination in one embodiment by querying an Identity System parameter field. If a real time status check is required, Identity System 40 retrieves the requested certificate's real time status from Certificate Authority 2084, as described above with reference to FIG. 59A. In some embodiments, Identity Server 40 also stores the retrieved real time certificate status and related validation information as shown in FIG. 59A. Identity Server 40 determines whether the certificate's real time status is valid (step 3430). If the status is valid, Identity Server 40 exports the certificate (step 3434). Otherwise, Identity Server 40 issues an error message to the user (step 3432).

Detailed Description Text (259):
If real time status checking was not required (step 3424), Identity Server 40 determines whether the user's export request falls within the Validation Interval for the certificate

(step 3428). As explained above, the Validation Interval is a window of time extending from the last time the certificate's real time status was retrieved. In one embodiment, the Validation Interval is one hour. In various embodiments, the Validation Interval has many different values. As the Validation Interval is reduced, the probability increases that the stored real time status for the certificate is still accurate. If the export request falls within the Validation Interval, Identity Server 40 exports the requested certificate (step 3434). Otherwise, Identity Server 40 issues an error message to the user (step 3432). By employing stored real time certificate status, Identity System 40 can supply real time status for large numbers of certificates. In one embodiment, the Validation Interval is zero for a certificate that is not valid--resulting in Identity Server 40 issuing an error message in response to the determination in step 3428.

Detailed Description Text (260):
FIG. 59C illustrates a sequence of steps executed by Identity Server 40 to display certificate information in one embodiment of the present invention. Identity Server 40 receives a user request via Web Server 20 to display a certificate in data store location 2082 (step 3450). Identity Server 40 determines whether certificate status is to be displayed along with the certificate (step 3452). In one implementation, Identity Server 40 makes this determination by querying a parameter field in the Identity System set by the Identity System administrator.

Detailed Description Text (261):
If certificate status is not required (step 3452), Identity Server 40 identifies the fields in the requested certificate that are to be displayed (step 3460). Identity Server 40 identifies these fields in one embodiment by querying a set of parameters in the Identity System that are programmed by the Identity System administrator. Identity System 40 then displays the identified fields from the certificate without any certificate status (step 3466).

Detailed Description Text (262):
If certificate status is required (step 3452), Identity Server 40 determines whether a real time certificate status check is required (step 3454). Identity Server 40 makes this determination in one implementation by querying an Identity System parameter field. If a real time status check is required, Identity Server 40 retrieves a new real time status for the certificate (step 3456), as described above with reference to FIG. 59A. In some implementations, Identity Server 40 also stores the status and validation information as shown in FIG. 59A. If a real time status check is not required (step 3454), Identity Server 40 retrieves previously obtained real time status that is stored in the Identity System for the certificate (step 3458).

Detailed Description Text (265):
The discussions above regarding workflows, groups, communication between Identity Servers, etc., primarily pertain to managing and using the Identity System. As stated above, the Identity System manages identity profiles. These identity profiles are used, among other things, to authenticate users and to authorize users to access resources. The Access System has primary responsibility for providing authentication and authorization services. In one embodiment, authentication and authorization services are performed based on using identity profiles with authentication and authorization rules. These authentication and authorization rules are associated with policy domains and policies, as described above.

Detailed Description Text (292):
In step 2556, the method determines whether the user is authorized to access the requested resource. If the user is authorized (step 2590), the method proceeds to step 2592. Otherwise, the unsuccessful authorization is logged in step 2596. After step 2596, the method performs authorization failure actions (step 2598) and Web Gate 28 denies the user access to the requested resource. If authorization is successful (step 2590), then the successful authorization of the user is logged in step 2592. Authorization success actions are performed in step 2594. The user is granted access to the requested resource in step 2595. In one embodiment of step 2595, some or all of HTTP request information is provided to the resource. In one or more scenarios, the resource being accessed is the Identity System.

CLAIMS: